```
#include<iostream>
#include<mpi.h>

int main(int argc, char* argv[]){

        MPI_Init(&argc, &argv);

        int rank, size;
        MPI_Comm_size(MPI_COMM_WORLD, &size);
        MPI_Comm_rank(MPI_COMM_WORLD, &rank);

        .. ;

        MPI_Finalize();
        return 0;
}

// Send/Receive (point-to-point)
MPI_Send(&n, 1, MPI_DOUBLE, dest, 0, MPI_COMM_WORLD);
MPI_Recv(&n, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);

// Broadcast (collective)
MPI_Bcast(&n, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);

// Reduce/Allreduce (MPI_MAX, MPI_MIN, MPI_SUM, MPI_PROD)
MPI_Reduce(&local_value, &total, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
MPI_Allreduce(MPI_IN_PLACE, &value, 1, MPI_DOUBLE, MPI_MIN, MPI_COMM_WORLD);

// Cyclic partitioning
for(size_t i=rank; i<v.size(); i+=size){ .. }

// Block partitioning: Scatter/Gather
MPI_Scatter(global.data(), local_n, MPI_DOUBLE, local.data(), local_n, MPI_DOUBLE, 0, MPI_COMM_WORLD);
MPI_Gather(local.data(), local_n, MPI_DOUBLE, global.data(), local_n, MPI_DOUBLE, 0, MPI_COMM_WORLD);
MPI_AllGather(local.data(), local_n, MPI_DOUBLE, global.data(), local_n, MPI_DOUBLE, MPI_COMM_WORLD);

// Read_vector/Print_vector
vector<double> x = mpi::read_vector(n, "x", MPI_COMM_WORLD);
vector<double> y = mpi::read_vector(n, "y", MPI_COMM_WORLD);
vector<double> z = x+y;
mpi::print_vector(z, n, "The global vector is: ", MPI_COMM_WORLD);

//-------------------------------------------------------------------------------------------------------------------

$ mpicxx --std=c++11 file_name.cc
$ mpiexec -np=4 a.out

// Name specification
$ mpicxx --std=c++11 file_name.cc -o exe_name
$ mpiexec -np=4 exe_name

// More files
$ mpicxx --std=c++11 file_name.cc file1.cc file2.cc -o exe_name
$ mpiexec -np=4 exe_name

// Arguments
$ mpicxx --std=c++11 file_name.cc -o exe_name
$ mpiexec -np=4 exe_name text1 text2              // argc=3, argv[1]="text1", argv[2]="text2"

// Standard input (only rank_0 has "std::cin >> a >> b >> c;")
$ mpicxx --std=c++11 file_name.cc -o exe_name
$ mpiexec -np=4 exe_name
5 6 7                                             // a=5, b=6, c=7

// Standard input with saved input and output
$ vim input_short
$ mpicxx --std=c++11 file_name.cc -o exe_name
$ mpiexec -np=4 exe_name < input_short > output_short

// User passed informations
$ mpicxx --std=c++11 filne_name.cc -o exe_name
$ mpiexec -np=4 exe_name 5 6 7
// ..
if (argc==4){
        unsigned var1 = std::stoul(argv[1]);
        unsigned var2 = std::stoul(argv[2]);
        int      var3 = std::stoi(argv[3]);
}
```